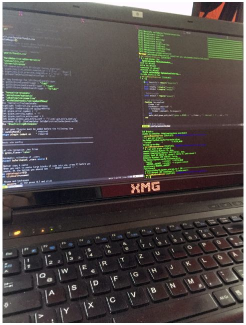
C introduction

Basic program structure

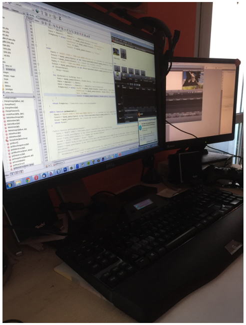
Richard Mörbitz, Manuel Thieme

Contents

OS's you may use



Linux

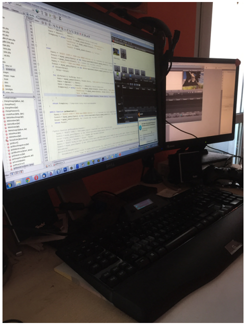


Windows

Mac OS X

OS’s you may use

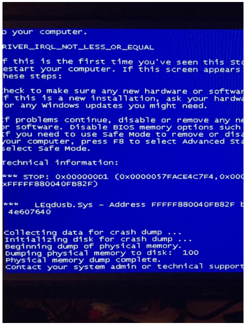
Linux
recommended


Windows

Mac OS X

OS's you may use

Linux
recommended



```
o your computer.

RIVER_IRQL_NOT_LESS_OR_EQUAL

f this is the first time you've seen this Sto
estart your computer. If this screen appears
hese steps:

heck to make sure any new hardware or softwar
f this is a new installation, ask your hardwa
or any Windows updates you might need.

f problems continue, disable or remove any ne
r software. Disable BIOS memory options such
f you need to use Safe Mode to remove or disa
your computer, press F8 to select Advanced Sta
elect Safe Mode.

Technical information:

*** STOP: 0x000000D1 (0x0000057FACE4C7F4,0x000
xFFFFF880040FB82F)


***   LEqdUsb.Sys - Address FFFFF880040FB82F b
 4e607640


Collecting data for crash dump ...
Initializing disk for crash dump ...
Beginning dump of physical memory.
Dumping physical memory to disk:  100
Physical memory dump complete.
Contact your system admin or technical support
```

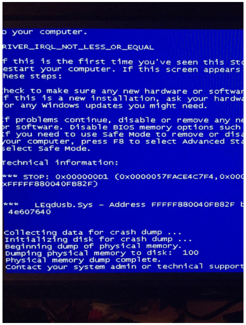
Windows
supported

Mac OS X

OS's you may use

Linux

recommended



```
o your computer.

RIVER_IRQL_NOT_LESS_OR_EQUAL

f this is the first time you've seen this St
estart your computer. If this screen appears
hese steps:

heck to make sure any new hardware or softwar
f this is a new installation, ask your hardwa
or any Windows updates you might need.

f problems continue, disable or remove any ne
r software. Disable BIOS memory options such
f you need to use Safe Mode to remove or disa
your computer, press F8 to select Advanced Sta
elect Safe Mode.

echnical information:

*** STOP: 0x000000D1 (0x0000057FACE4C7F4,0x000
xFFFFF880040FB82F)


***   LEqdusb.sys - Address FFFFF880040FB82F b
 4e607640


Collecting data for crash dump ...
Initializing disk for crash dump ...
Beginning dump of physical memory.
Dumping physical memory to disk:  100
Physical memory dump complete.
Contact your system admin or technical support
```

Windows

supported

~~Mac OS~~

Installing gcc on Linux

Ubuntu / Debian:

```
$ sudo apt-get install gcc
```

Arch:

```
$ sudo pacman -S gcc
```

... and you're done ;-)

cygwin

- Download installer from *https://cygwin.com/install.html*
- Run it
  - "Install from Internet"
  - Choose your installation path
  - Choose path for installation files
  - "Direct Connection"
  - Choose a mirror
  - Important software already is selected
  - Optional: powerful editor "vim" in *Editors*
  - Watching loading bars...
  - ???
  - Profit!
- Use cygwin-console like a linux terminal

The first program

- Create a new file named **main.c**.
- Open it in your text editor of trust.
- Fill it as follows:

```c
#include <stdio.h>

int main(int argc, char *argv[]) {
    printf("Hello World!\n");
    /* Print "Hello World!" on the
       command line */
    return 0;
}
```

From source to bits

Source code

$\Downarrow$

```
$ gcc main.c
```

(Preprocessing, compiling, assembling, linking)

$\Downarrow$

Executable program

Linux (**a.out**)

```
$ ./a.out
```

Windows (**a.exe**)

```
$ ./a.exe
```

A basic program

```c
#include <stdio.h>

int main(int argc, char *argv[]) {

    printf("Hello World!\n");
    /* Print "Hello World!" on the
       command line */

    return 0;
}
```

} Preprocessor statements

} Main function

Preprocessor statements

- Processed before compilation
- Have their own language, start with a #

```
#include <stdio.h>
```

- Includes the **standard input/output library** (needed for printf, which is defined there)

- Can also be used to define constants and much more, e.g.

```
#define THE_ANSWER 42
```

The main function

- Basic function
- Exists **exactly once** per program
- Called on program start

```
int main(int argc, char *argv[]) {
```

- As a function, *main()* takes parameters
- Get used to *argc* and *argv*, they will be explained later
- '{' marks the start of the main function scope

The main function scope

- Contains all program statements
- They are processed from top to bottom

```
    return 0;
}
```

- Last statement, ends main function (and thus the whole program)
- *0* tells the OS that everything went right
- '}' marks the end of the main function scope

Statements

- Instructions for the computer
- End with a ; (semicolon)

```
    printf("Hello World!\n");
```

- There is the empty statement:

```
    ;
```

- All statements are located in function blocks

Comments

- Information for the programmer, cut out before compilation

Single line comments:

```
    // Print "Hello World!" on the command line
```

Block comments (mutli-line):

```
    /* Print "Hello World!"
       on the command line */
```

Better use of block comments:

```
    /*
     * Print "Hello World!"
     * on the command line
     */
```

A few words on style

- There can be multiple statements on one line
- Intendation is not nessessary at all

A few words on style

- There can be multiple statements on one line
- Intendation is not nessessary at all
- **But**...

```
#include <stdio.h>
int
main     (int  argc,  char *argv[]){printf("Hello  World!\n");
         //  Prints
/*"Hello  World!"             */
         return  0;}
```

Much more enjoyable

- Put each statement on a single line
- Intend every statement in the main function by one tab / 4 *spaces*
- Use */* ... */* rather than *// ...*
- Write the main function arguments directly behind *main*
- Leave a *space* between the closing ')' and the opening '{'